|  |  |
| --- | --- |
| **МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**  **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  **«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**  **(РУТ (МИИТ)**  **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ** | |
| **Лабораторная работа №2**  **по дисциплине «Методы программирования»**  **«Конструктор ГПИ»** | |
|  | Выполнил:  Студент группы ТКИ-341  Пряхин И.Д. Кокарев П.В. |
|  | Проверил:  Доцент кафедры УиЗИ к.т.н., с.н.с  Сафронов А.И. |
| Москва 2024 | |

**Оглавление**

[1. Цель работы 3](#__RefHeading___1)

[2. Формулировка задачи 3](#__RefHeading___2)

[3. Индивидуальная задача 4](#__RefHeading___3)

[3.1. Размещение основных элементов: 5](#__RefHeading___4)

[3.2. Индивидуальное изображение: 5](#__RefHeading___5)

[3.3. Индивидуальные элементы управления: 5](#__RefHeading___6)

[3.4. Диаграммы классов, входящих в состав решения. 6](#__RefHeading___7)

[4. Сеть Петри 7](#__RefHeading___8)

[5. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (меток) 9](#__RefHeading___9)

[6. Листинг (код) программы 12](#__RefHeading___10)

[6.1. Листинг модуля Programma 12](#__RefHeading___11)

[6.2. Листинг модуля MainMenu 22](#__RefHeading___12)

[7. Ожидаемые результаты 24](#__RefHeading___13)

[8. Пример работы программы 25](#__RefHeading___14)

[9. Вывод 25](#__RefHeading___15)

1. Цель работы

Закрепить навыки разработки визуального пользовательского интерфейса, освоить работу с текстовыми файлами и кодировкой в среде *Microsoft* *Visual Studio*, научиться реализовывать настройку множественных состояний объектов посредством управления компонентами со внутренней индексацией

1. Формулировка задачи

В интегрированной среде разработки *Microsoft* *Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой экранную форму, содержащую главное меню, позволяющее:

* Начать работу с приложением.
* Прервать работу приложения.
* Предоставить пользователю справочную информацию о работе с приложением.

Сама программа должна реализовывать вывод в графический элемент управления (например, *PictureBox*) главной экранной формы плоскостную геометрическую фигуру, выбираемую пользователем из списка (вид списка \* задаётся вариантом индивидуального задания). Список должен обязательно содержать следующие пункты:

* «Квадрат»,
* «Прямоугольный треугольник»,
* «Эллипс»,
* «Равнобедренный треугольник»,
* «Круг»,
* «Равносторонний треугольник»,
* «Окружность»,
* «Ромб»,
* «Трапеция»,
* «Параллелограмм»,

«Прямоугольник».

1. Индивидуальная задача

* Заголовок экранной формы должен содержать надпись вида: «Задание №2 выполнил: [Фамилия И.О. авторов]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг]».
* Дата выполнения проставляется в момент, когда программа считается законченной и по ней можно готовить итоговый отчёт о выполнении работы.
* Заменить стандартный курсор экранной формы со «стрелки» на «руку».
* Справочная информация должна быть вызвана в дочерней экранной форме и считана в статическое по размеру текстовое поле (*TextBox*) с выставленным запретом на редактирование текстовой информации
* Заменить цвет подложки на Desktop.
* Подобрать читаемый цвет текста к схеме (руководствоваться принципами разработки эргономичного графического пользовательского интерфейса).
* Исходное состояние всех элементов, расположенных на главной экранной форме, должно быть настроено через перечень параметров этих элементов.
* В качестве исходного состояния принимается заранее известная и заполненная элементами структура списков, все элементы экранной формы за исключением главного меню находятся либо в недоступном состоянии (.Enabled = false), либо в невидимом состоянии (.Visible = false).
* Пункт меню «Начало работы с приложением» должен реализовывать активацию доступа пользователя к элементам или отображение элементов на экранной форме для пользователя.
* Все элементы программы должны носить значащие имена переменных, в которых отражено существо этих элементов, например, главная экранная форма – frmMain, ярлык – lblHelp, комбинированный список – cmbFigures и т.д.

## 3.1. Размещение основных элементов:

11, 7, 10, 6, 9, 3, 4, 8, 5, 2, 1.

## 3.2. Индивидуальное изображение:

Первая четверть декартовой системы координат (оси со стрелками, текстом {0}на пересечении,но без засечек).

## 3.3. Индивидуальные элементы управления:

Radiobutton.

## 3.4. Диаграммы классов, входящих в состав решения.

|  |
| --- |
|  |
| 1. Диаграмма классов |

1. Сеть Петри

|  |
| --- |
|  |
| 1. Условные обозначения |

|  |
| --- |
|  |
| 1. Полная сеть Петри |

1. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (меток)

|  |
| --- |
|  |
| 1. Запуск программы |

|  |
| --- |
|  |
| 1. Выход из приложения |

|  |
| --- |
|  |
| 1. Вызов справки |

|  |
| --- |
|  |
| 1. Окно справки |

|  |
| --- |
|  |
| 1. Меню выбора фигуры |

|  |
| --- |
|  |
| 1. Кнопка выхода из приложения |

1. Листинг (код) программы

## 6.1. Листинг модуля Programma

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab2

{

public partial class Programma :Form

{

MainMenu frm1 = new MainMenu();

private Graphics graphics;

private ShapeType selectedShape; //переменная "selectedShape" хранит выбранный тип фигуры

private bool isDrawing; //Для нажатия мышки

private List<Shape> shapes = new List<Shape>(); // для хранения инф. о фигурах, которые использовал пользователь

//Dictionary<ShapeType, Size> initialSizes = new Dictionary<ShapeType, Size>(); // Для начального размера фигур

public Programma()

{

InitializeComponent();

graphics = pictureBox1.CreateGraphics();

selectedShape = ShapeType.None;

isDrawing = false; // Для мышки

}

private void ExitB\_Click(object sender, EventArgs e)

{

this.Close();

//MainMenu frm1 = new MainMenu();

Application.Exit();

frm1.Show();

frm1.Close();

}

public class Shape

{

public ShapeType Type { get; set; }

public Point Location { get; set; }

public Shape(ShapeType type, Point location)

{

Type = type;

Location = location;

}

}

private void DrawAllShapes()

{

ClearPictureBox();

foreach (var shape in shapes)

{

DrawShape(shape.Type, shape.Location);

}

}

public enum ShapeType

{

None,

Circle,

Rectangle,

Square,

Ellipse,

RightTriangle,

IsoscelesTriangle,

EquilateralTriangle,

Rhombus,

Trapezoid,

Parallelogram,

FilledCircle,

coordinates

// Добавьте другие типы фигур

}

private void DrawShape(ShapeType shapeType, Point location)

{

//ClearPictureBox();

switch (shapeType)

{

case ShapeType.Circle:

graphics.DrawEllipse(Pens.White, location.X - 50, location.Y - 50, 100, 100);

break;

// Окружность

case ShapeType.Rectangle:

graphics.DrawRectangle(Pens.White, location.X - 50, location.Y - 50, 200, 100);

// Прямоугольник

break;

case ShapeType.Square:

graphics.DrawRectangle(Pens.White, location.X - 50, location.Y - 50, 100, 100);

// Квадрат

break;

case ShapeType.Ellipse:

graphics.DrawEllipse(Pens.White, location.X - 50, location.Y - 30, 100, 60);

// Эллипс

break;

case ShapeType.RightTriangle:

Point[] trianglePoints = { new Point(location.X - 50, location.Y + 50), new Point(location.X + 50, location.Y + 50), new Point(location.X - 50, location.Y - 50) };

graphics.DrawPolygon(Pens.White, trianglePoints);

// Прямоугольный треугольник

break;

case ShapeType.IsoscelesTriangle:

Point[] isoscelesTrianglePoints = { new Point(location.X - 50, location.Y + 50), new Point(location.X + 50, location.Y + 50), new Point(location.X, location.Y - 50) };

graphics.DrawPolygon(Pens.White, isoscelesTrianglePoints);

// Равнобедренный треугольник

break;

case ShapeType.EquilateralTriangle:

Point[] equilateralTrianglePoints = { new Point(location.X, location.Y - 50), new Point(location.X + 43, location.Y + 25), new Point(location.X - 43, location.Y + 25) };

graphics.DrawPolygon(Pens.White, equilateralTrianglePoints);

// Равносторонний треугольник

break;

case ShapeType.Rhombus:

Point[] rhombusPoints = { new Point(location.X, location.Y - 50), new Point(location.X + 50, location.Y), new Point(location.X, location.Y + 50), new Point(location.X - 50, location.Y) };

graphics.DrawPolygon(Pens.White, rhombusPoints);

// Ромб

break;

case ShapeType.Trapezoid:

Point[] invertedTrapezoidPoints = { new Point(location.X - 60, location.Y + 40), new Point(location.X + 60, location.Y + 40), new Point(location.X + 40, location.Y - 40), new Point(location.X - 40, location.Y - 40) };

graphics.DrawPolygon(Pens.White, invertedTrapezoidPoints);

// Трапеция

break;

case ShapeType.Parallelogram:

Point[] parallelogramPoints = { new Point(location.X - 40, location.Y + 40), new Point(location.X + 20, location.Y + 40), new Point(location.X + 40, location.Y - 40), new Point(location.X - 20, location.Y - 40) };

graphics.DrawPolygon(Pens.White, parallelogramPoints);

// Параллелограмм

break;

case ShapeType.FilledCircle:

graphics.FillEllipse(Brushes.White, location.X - 50, location.Y - 50, 100, 100);

// круг

break;

case ShapeType.coordinates:

Point centerPoint = new Point(location.X, location.Y);

//рисуем оси x и y

graphics.DrawLine(Pens.White, location.X, location.Y - 5, location.X + 40, location.Y - 5);

graphics.DrawLine(Pens.White, location.X, location.Y - 5, location.X, location.Y - 50);

//стрелочка по оси Х

graphics.DrawLine(Pens.White, location.X + 40, location.Y + -5, location.X + 35, location.Y + -10);

graphics.DrawLine(Pens.White, location.X + 35, location.Y + -10, location.X + 35, location.Y );

graphics.DrawLine(Pens.White, location.X + 35, location.Y , location.X + 40, location.Y -5);

//стрелочка по оси Y

graphics.DrawLine(Pens.White, location.X, location.Y -50, location.X + 5, location.Y -45);

graphics.DrawLine(Pens.White, location.X + 5, location.Y -45, location.X - 5, location.Y -45);

graphics.DrawLine(Pens.White, location.X - 5, location.Y -45, location.X, location.Y -50);

graphics.DrawString("0", new Font("Times New Roman",8),Brushes.White, centerPoint);

break;

}

}

private void ClearPictureBox()

{

graphics.Clear(SystemColors.Desktop);

}

private void RB1\_CheckedChanged(object sender, EventArgs e)

{

if (RB1.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Rectangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB2\_CheckedChanged(object sender, EventArgs e)

{

if (RB2.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Circle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB3\_CheckedChanged(object sender, EventArgs e)

{

if (RB3.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Parallelogram;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB4\_CheckedChanged(object sender, EventArgs e)

{

if (RB4.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.EquilateralTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB5\_CheckedChanged(object sender, EventArgs e)

{

if (RB5.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Trapezoid;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB6\_CheckedChanged(object sender, EventArgs e)

{

if (RB6.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Ellipse;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB7\_CheckedChanged(object sender, EventArgs e)

{

if (RB7.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.IsoscelesTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB8\_CheckedChanged(object sender, EventArgs e)

{

if (RB8.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Rhombus;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB9\_CheckedChanged(object sender, EventArgs e)

{

if (RB9.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.FilledCircle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB10\_CheckedChanged(object sender, EventArgs e)

{

if (RB10.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.RightTriangle;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

private void RB11\_CheckedChanged(object sender, EventArgs e)

{

if (RB11.Checked)

{

ClearPictureBox();

shapes.Clear();

selectedShape = ShapeType.Square;

isDrawing = true;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

/////

private void ClearB\_Click(object sender, EventArgs e)

{

ClearPictureBox();

shapes.Clear();

}

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

if (isDrawing == true)

{

shapes.Add(new Shape(selectedShape, e.Location));

DrawAllShapes();

}

}

bool models = false;

private void CBcoordinates\_CheckedChanged(object sender, EventArgs e)

{

if (CBcoordinates.Checked)

{

ClearPictureBox();

shapes.Clear();

isDrawing = true;

selectedShape = ShapeType.coordinates;

}

else

{

selectedShape = ShapeType.None;

isDrawing = false;

}

}

}

}

## 6.2. Листинг модуля MainMenu

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab2

{

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

}

private void StopB\_Click(object sender, EventArgs e)

{

Application.Exit();

}

//Пробный варинт

private void frm2\_FormClosed(object sender, FormClosedEventArgs e)

{

this.Show();

}

//

private void StartB\_Click(object sender, EventArgs e)

{

Programma frm2 = new Programma();

//frm2.TopMost = true;

frm2.WindowState = FormWindowState.Maximized;

frm2.FormClosed += new FormClosedEventHandler(frm2\_FormClosed);

this.Hide();

frm2.ShowDialog();

}

public bool checkMain = false;

private void SpravB\_Click(object sender, EventArgs e)

{

Help frm3 = new Help();

//Help frm3 = Help.GetInstance();

if (!checkMain )

{

checkMain = true;

frm3.Show();

//frm3.ShowDialog(this);

}

else if (checkMain)

{

checkMain = false;

frm3.Focus();

}

//this.Hide();

}

}

}

1. Ожидаемые результаты

|  |
| --- |
|  |
| 1. Примеры работы программы |

1. Пример работы программы

После последних правок при выборе любой фигуры (модели) происходит очистка листа. На рисунках ниже приведены примеры черчения каждой модели из списка.

|  |
| --- |
|  |
| 1. Примеры работы программы |

1. Вывод

В процессе выполнения данной лабораторной работы были получены навыки разработки визуального пользовательского интерфейса, была освоена работа с текстовыми файлами и кодировкой в среде Microsoft Visual Studio, научились реализовывать настройку множественных состояний объектов посредством управления компонентами со внутренней индексацией.